資料探勘 Project1 報告

資訊三 E44051259 陳冠霖

前言：

在這個資訊量大爆炸的時代，我們常常會拿到很多大量的資料，而怎麼讓這些資料變成有用的東西，是資料探勘主要的工作，但往往有時候我們要從一堆數據中去找出出現頻率較高的資料是一件困難且費時的事情，所以今天這份報告主要要比較三種不同的演算法在處理不同筆數或不同item數的資料時，執行速度上的差異。

演算法介紹：

一：暴力法

沒有任何的技巧，就是將所有的condidate窮舉出來，之後再逐筆與transation log 比較，如果condidate是transation的subset則condidate加二：Apriori

一樣依然要先建立好condidate，但這個演算法的想法是，因為我們有一個min\_sup，而我們從condidate數量為1的開始比較，然後刪除sup小於min\_sup的，然後依序做下去，這樣的好處是，可以減少要比較的condidate數量

三：FP-growth

這是三個演算法中最複雜的一個演算法，首先我們必須將以item的出現頻率，去排序每一筆transation log，而當中刪除頻率小於min\_sup的item，然後再用排序好的transation log去建立FP-tree，最後我們從這個建好的tree中的leaf去求出可能的path，並去求path中的組合的condidate數量。其中這個path就是所謂的condidate pattern

資料集：

我們準備了五種資料集：

1. 從kaggle上找到的買賣物品資料集：有約莫10000筆, item數為：95
2. 從IBM工具產生的資料 ：有約莫10000筆, item數為：100
3. 從IBM工具產生的資料 ：有約莫10000筆, item數為：10
4. 從IBM工具產生的資料 ：有約莫1000 筆, item數為：100
5. 從IBM工具產生的資料 ：有約莫1000 筆, item數為：10

比較結果：

1. 從kaggle上找到的買賣物品資料集：有約莫10000筆, item數為：95

min\_sup=19

|  |  |  |  |
| --- | --- | --- | --- |
| 演算法 | 暴力法 | Apriori | FP-growth |
| 執行時間 | >2 hours | 54s | 0.78s |
|  |  |  |  |

速度 FP-growth >> Apriori >> 暴力法

2. 從IBM工具產生的資料 ：有約莫10000筆, item數為：100

min\_sup=834

|  |  |  |  |
| --- | --- | --- | --- |
| 演算法 | 暴力法 | Apriori | FP-growth |
| 執行時間 | >2 hours | 0.9s | 1.2s |

速度 Apriori ~= FP-growth >> 暴力法

3. 從IBM工具產生的資料 ：有約莫10000筆, item數為：10

min\_sup=834

|  |  |  |  |
| --- | --- | --- | --- |
| 演算法 | 暴力法 | Apriori | FP-growth |
| 執行時間 | 7.2s | 0.9s | 0.12s |

速度 Apriori ~= FP-growth > 暴力法

4. 從IBM工具產生的資料 ：有約莫1000 筆, item數為：100

min\_sup=84

|  |  |  |  |
| --- | --- | --- | --- |
| 演算法 | 暴力法 | Apriori | FP-growth |
| 執行時間 | >2 hours | 0.096s | 0.186s |

速度 Apriori ~= FP-growth >> 暴力法

5. 從IBM工具產生的資料 ：有約莫1000 筆, item數為：10

min\_sup=81

|  |  |  |  |
| --- | --- | --- | --- |
| 演算法 | 暴力法 | Apriori | FP-growth |
| 執行時間 | 0.6s | 0.12s | 0.048s |

速度 FP-growth > Apriori > 暴力法

結論：

由上面的實驗結果我們可以將資料的類型主要分成兩部分來討論：

分別是：min\_sup的值以及item的數量，這兩個數值是主要引響執行速度的原因。

當min\_sup變小時，代表要計算的condidate變多了，這時候FP-growth非常明顯的大贏Apriori，這點可以由資料集1和2中看得出來。

而當item數量變多時，例如從10變成100時暴力法運算所需要的時間是成指數型成長的，這時候只能使用另外兩種演算法了

最後資料的筆數對於演算法個體之間的差異影響不大。